IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	§	**ATTORNEY DOCKET NO.: AT9-99-505**
GUY LYNN GUTHRIE, ET AL.	§	
Serial No.: **09/588,508**	§	Examiner: **LI, AIMEE J.**
Filed: **JUNE 6, 2000**	§	Art Unit: **2183**
For: **SYSTEM AND METHOD FOR ENABLING WEAK CONSISTENT STORAGE ADVANTAGE TO A FIRMLY CONSISTENT STORAGE ARCHITECTURE**	§	

RESPONSE TO NOTICE OF NON-COMPLIANT APPEAL BRIEF UNDER 37 C.F.R. 41.37

Mail Stop Appeal Briefs - Patents
Commissioner for Patents
Washington, D.C. 20231

Sir:

This Appeal Brief is submitted in response to a Notice of Non-Compliant Appeal Brief for Appeal Brief filed on September 16, 2004. No fee is required to file this Compliant Appeal Brief as the fee for filing the original Appeal Brief was paid at submission. However, should any fees be required to file this Compliant Appeal Brief, please charge that fee, as well as any additional required fees, to IBM Deposit Account No. 09-0447. A one month extension of time is required and hereby requested. A check in the amount of $120.00 is enclosed to cover the one month extension of time.

Certificate of Transmission/Mailing

I hereby certify that this correspondence is being facsimile transmitted to the USPTO at 703-872-9306 or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450 on the date shown below.

Typed or Printed Name: Shenise Ramdeen *Date:* January 27, 2005 *Signature:* S Ramdeen

02/03/2005 MAHMED1 00000013 09588508

01 FC:1251 120.00 OP

REAL PARTY IN INTEREST

The real party in interest in the present Appeal is International Business Machines Corporation, the Assignee of the present application as evidenced by the Assignment recorded at reel 010890 and frame 0488 *et. seq.*

RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, the Appellants' legal representative, or assignee, which directly affect or would be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-11 stand finally rejected by the Examiner as noted in the Advisory Action dated July 2, 2004.

STATUS OF AMENDMENTS

Appellants' Amendment B, filed on June 3, 2004, was not entered by the Examiner.

SUMMARY OF THE CLAIMED SUBJECT MATTER

Appellants' invention provides ordered completion of memory access requests issued by an in-order processor that is operationally connected to a memory subsystem that allows the completion of memory access requests in a weakly consistent order. The invention enables processor-issued memory access requests to be processed in any order at the memory subsystem but "completed" at the processor-level in the correct program order. A controller of the processor automatically generates and places a barrier operation (i.e., a memory scheduling operation that forces an ordered completion of memory access operations at the memory and returns an acknowledgment when the preceding memory access operation(s) is completed) on the system bus after each memory access request is issued from the processor (*see* Figure 6C). The memory access requests complete in any order (i.e., out-of-order) at the memory subsystem. However, because of the barrier instruction issued after each request, the values/completion signals returned to the processor are forced to be sent to the processor's execution units in the

order the requests were issued (i.e., in order) irrespective of the order they would have been processed at the "out-of-order" memory subsystem (*see* page 30, line 17 - page 31, line 21).

The existence of a preceding barrier operation on the system bus is ignored from a processor-dispatch perspective whenever a new load request instruction appears in the processing sequence. Thus, unlike conventional implementations, previously-issued barrier operations that have not yet completed (i.e., returned an acknowledgment of completion) are not considered an absolute bar to issuing additional memory access requests. A subsequent load request in the instruction sequence is "speculatively" issued with respect to the pending barrier operation (*see* page 15, ll 21-32; page 18, ll 6-9; and page 20 ll 7-15). The data returned by that subsequent load request is, however, not allowed to be utilized by the processor until an acknowledgement of completion is received for each/all previously-issued barrier instruction. This also enables the corresponding processor-level operations to complete processing (from a processor-perspective) in the order in which the memory access requests were issued.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. The Examiner's rejection of Claims 1-2, 5-6 and 9 under 35 U.S.C. § 102(e) as being anticipated by *Sproull* (U.S. Patent No. 6,038,646) is to be reviewed on Appeal;

B. The Examiner's rejection of Claims 3-4, 7-8 and 10-11 under 35 U.S.C. § 103(a) as being unpatentable over *Sproull* in view of *Karp, et al.* (U.S. Patent No. 6,321,328) is to be reviewed on Appeal.

ARGUMENT

A. Examiner's rejection of Claims 1-2, 5-6 and 9 under 35 U.S.C. § 102(e), as being anticipated by *Sproull*, is not well founded and should be reversed.

Exemplary Claims 1 and 9

Exemplary Claim 1 of Appellants' invention recites the following key component and associated functionality:

"a controller associated with said processor that forwards said memory access requests to said memory system and which **automatically places a barrier operation on said interconnect following each issuance of a memory access request to said memory system,** wherein said barrier operation indicates a need to complete the data operations associated with the memory access requests in program order from the perspective of the processor" (emphases added).

Notably, corresponding Claim 9 also recites:
"in response to receipt of said memory access instruction, **generating** a memory access request **and a barrier operation; automatically initiating said barrier operation** after said memory access request is issued to a memory system." Further, Claim 2 recites: "wherein said **controller** includes **means for creating said barrier operations,**" (emphases added).

Appellants' claims thus enable every memory access operation to be completed in-order from the processor perspective and provide an instruction-by-instruction barrier control feature. The invention is implemented within instruction code that does not contain any barrier operations (i.e., in-order processors do not routinely use barrier operations) and actually generates the barrier operation on the interconnect following issuance of **each** memory access operation. To enable the processor-level ordering of each issued request to the out-of-order memory subsystem, the controller generates a barrier operation and placed the barrier operation on the system bus to force the processor-level "completion" of the issued load requests in processor order.

Sproull does not teach the above recited claim features and therefore does not anticipate the above list of Appellants' claims. Specifically, *Sproull* does not teach "automatically places a barrier operation ... following **each issuance** of a memory access request to said memory system." *Sproull* provides a memory interface (between a processor and memory subsystem that is capable of multiple concurrent transactions (read and write operations and barrier operations that signal the non-reorder-ability of operations) or accesses (Abstract). *Sproull* further describes and illustrates an instruction stream divided up into **sets/groups** of memory access instructions

separated by a barrier operation, when a memory instruction in a first set of instructions references the same memory address as an instruction in the second set (see Fig. 3; *see also,* col. 8, ll 43 – 49). At col. 8, line 58-61, *Sproull* clearly states that "the processor need not introduce barrier requests to enforce ordering of requests when one of the requests has already been acknowledged by the memory subsystem." *Sproull* also allows the memory access instructions within a set to complete in any order within the set (*see*, col. 8, ll 13-25).

Further, as is clearly illustrated by the instruction stream of *Sproull's* Figure 3, a set of instructions may contain a series of memory access request that are issued by *Sproull*'s processor without any barrier operation being placed after each issuance. In fact, the generation and placement of the MEMIBAR operations is not an automatic function performed by a controller. Rather it is a more complex process that requires an actual determination that two memory access operations are targeting the same memory address (i.e., an address conflict) (*see* col. 9, ll 4-7). The instruction stream is then separated into sets by placing barrier operations at these pre-determined or pre-calculated locations to separate particular memory access instructions within the issuing sequence with existing address conflicts. The barrier operation(s) in *Sproull* are inserted with the functional objective of separating memory access instructions and are inserted only when a next memory access instruction references the same memory address as a previous instruction in the instruction stream. *Sproull* is solely concerned with protecting a particular identified memory location when there are multiple access requests addressed thereto.

Examiner's analysis indicates that Examiner misses a key feature of Appellants' claimed invention, which is the placement of a barrier operation after each memory access request to order the operations to complete in the processor's order. This in-order processing requirement and implementation required to achieve the in-order processing even when accessing an out-of order memory subsystem is not taught nor suggested by *Sproull's* attempts to avoid address conflicts between to memory access instructions. That is, automatically placing a barrier operation after each memory access request is inherently distinguishable from placing a barrier operation between sets of instructions when an instruction is a first set targets a same address as another instruction in the second set.

Thus, it is clear that *Sproull* does not teach "a controller... that ...automatically places a barrier operation on the interconnect following each issuance of a memory access request to said memory system." It is also clear that *Sproull* does not teach or contemplate actual generation of a new barrier operation for each memory access request that is issued to the memory subsystem.

The standard for a § 102 rejection requires that the reference teach each element recited in the claims set forth within the invention. As clearly outlined above, *Sproull* fails to meet this standard and therefore do not anticipate Appellants' Group I claims. Examiner's rejection of the Group I claims as being anticipated by *Sproull* is therefore not well founded and should be reversed.

B. Examiner's rejection of Claims 3-4, 7-8 and 10-11 under 35 U.S.C. § 103(a) as being unpatentable over *Sproull* in view of *Karp* is not well founded and should be reversed.

Sproull is utilized to support the rejections of the general features of the Group II claim, while *Karp* is utilized by the Examiner to support the 103 rejection of the "speculative" issuing feature recited by Appellants' exemplary Claim 3. Since the Group II claims depend on respective independent claims of Group I, the above arguments rendering the group I claims allowable over *Sproull* necessarily overcomes the rejections of the Group II claims.

Exemplary Claim 3

Appellant's exemplary Claim 3 recites the following feature, which is not suggested by the combination of *Sproull* and *Karp*:

> "means for ignoring a pending barrier operation when a subsequent load request instruction appears in the instruction sequence; and ... speculatively issuing the subsequent load request to said memory system before the pending barrier operation is completed, wherein said subsequent load request is speculative because said subsequent load request is issued before a previous memory access request that may invalidate or change data retrieved by said load request completes within the memory system," (emphasis added).

Additionally, as defined within Appellants' specification and claims, Appellants' reference to "speculatively issuing" a load instruction refers to issuing a next load instruction before an acknowledgment of completion has been provided/received for a previously-issued barrier operation. Applicants' claims utilizes the term speculative in the context of "ignoring" the status of a previously issued barrier operation and issuing a subsequent load request to the interconnect before the barrier operation completes. Further, the exemplary claim expands this definition to include: "wherein said subsequent load request is speculative because said subsequent load request is issued before a previous memory access request that may invalidate or change data retrieved by said load request completes within the memory system."

Thus, while *Karp* mentions the term "speculative", *Karps* use of that term is inherently different and distinguishable from the use attributed to that term by Appellants' specification and claims. That is, *Karp* does not utilized that term to mean issuing load requests while a previously issued barrier operation has not been completed.

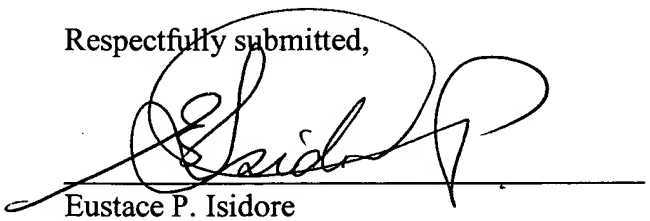
Karp describes a conventional speculative issuance of an instruction based on executing branch instructions, which involves later determination of whether the associated speculative branch paths are taken or not taken. *Karp* moves a load instruction up in the instruction sequence and the instruction is issued speculatively because "the compiler does not know if ... branch instructions will take a path away from the load instruction"(col. 1, ll 50-53). Those skilled in the art are familiar with this conventional use of the term speculative as utilized to refer to resolving branch instructions.

It is clear that Appellants' use of the term "speculative" is in a different context and provides a very different functionality than what is provided by *Karp*. It is also clear that *Karps'* use of that term is not suggestive of the specific functionality attributed to Appellants' speculative issuance of a load request before a previous barrier operation has completed. For these reasons, one skilled in the art would not find Applicants invention obvious in light of the combinations and the above claims are allowable. Examiner's rejection of the Group II claims is therefore not well founded and should be reversed.

CONCLUSION

Appellants have pointed out with specificity the manifest error in the Examiner's rejections, and the claim language which renders the invention patentable over *Sproull* and the combination of *Sproull* and *Karp*. Appellants, therefore, respectfully requested that this case be remanded to the Examiner with instructions to issue a Notice of Allowance with respect to all pending claims.

Respectfully submitted,



Eustace P. Isidore
Reg. No. 56,104
Dillon & Yudell LLP
8911 North Capital of Texas Highway
Suite 2110
Austin, Texas 78759
512.343.6116

ATTORNEY FOR APPELLANTS

APPENDIX

1. A data processing system comprising:

an interconnect;

an in-order processor that issues all memory access requests in program order, wherein said processor issues said memory access requests from an instruction sequence only in said program order and accepts data retrieved by a first and a second memory access request into its execution units only in said program order;

a memory system coupled to said processor which supports completing memory access requests in a weakly consistent order; and

a controller associated with said processor that forwards said memory access requests to said memory system and which automatically places a barrier operation on said interconnect following each issuance of a memory access request to said memory system, wherein said barrier operation indicates a need to complete the data operations associated with the memory access requests in program order from the perspective of the processor.

2. The data processing system of Claim 1, wherein said controller includes means for creating said barrier operations.

3. The data processing system of Claim 1, further comprising:

means for ignoring a pending barrier operation when a subsequent load request appears in the instruction sequence; and

means for speculatively issuing the subsequent load request to said memory system before the pending barrier operation is completed, wherein said subsequent load request is speculative because said subsequent load request is issued before a previous memory access request that may invalidate or change data retrieved by said load request completes within the memory system.

4. The data processing system of claim 3, wherein said controller includes means for allowing data returned by a speculatively issued load request to be utilized by said processor

only when an acknowledgment is received from all barrier operations pending when said load was issued.

5. An in-order processor comprising:

an instruction sequencing unit (ISU) that receives memory access instructions in program order;

a load store unit (LSU) including a controller that issues memory access requests associated with said memory access instructions to an interconnect that couples said processor to a memory system and wherein said controller includes means for creating barrier operations and wherein said controller automatically places a barrier operation on said interconnect in response to each issuance of a memory access request, wherein all said memory access requests are forwarded to memory in the program order.

6. The processor of Claim 5, wherein said controller includes means for creating said barrier operations.

7. The processor of Claim 5, wherein said controller includes:

means for ignoring a pending barrier operation when a subsequent load request appears in the instruction sequence; and

means for speculatively issuing the subsequent load request to said interconnect before the pending barrier operation is completed, wherein said subsequent load request is speculative because said subsequent load request is issued before a previous memory access request that may invalidate or change data retrieved by said load request completes within the memory system.

8. The data processing system of claim 7, wherein said controller includes means for allowing data returned by a speculatively issued load request to be utilized by said processor only when an acknowledgment is received from all barrier operations pending when said load was issued.

9. A method of processing instructions in a data processing system having a memory system, said method comprising the steps of:

receiving an instruction sequence at a processor in program order, said instruction sequence including at least a first and a second memory access instruction;

in response to receipt of said memory access instruction, generating a memory access request and a barrier operation;

automatically initiating said barrier operation after said memory access request is issued to a memory system; and

upon completion of said barrier operation, completing said first and said second memory access request in program order at said processor.

10. The method of Claim 9, wherein said second memory access request is a load request and said method further comprises:

ignoring a pending barrier operation of said first memory access request when a subsequent memory access request in the instruction sequence is a load request; and

speculatively issuing said load request to said memory system before the pending barrier operation is completed, wherein said load request is speculative because said load request is issued before a previous memory access request that may invalidate or change data retrieved by said load request completes within the memory system.

11. The method of Claim 10, further including the step of forwarding data returned by said speculatively issued load request to a register or execution unit of said processor, when an acknowledgment is received for said barrier operation.